cap10morgan / **clojure_dev_candidate_exercise.md**
Last active 16 days ago

Clojure dev candidate exercise

clojure_dev_candidate_exercise.md

TODO List

This assignment is designed to gauge how well you know or can pick up Clojure and some common libraries to build a simple web application.

You should spend no more than a week on it. In fact, it is due one week after it is assigned. You should also focus on depth rather than breadth. Do as much good work as you can on each step rather than trying to do a little work on more steps.

Since we want to evaluate all of these projects anonymously, you should *not* include any identifying data in the code (your name, city, email address, phone number, school, etc.). We will also not be able to give feedback on it until the recruitment process is over.

However, you *may* ask questions of the Democracy Works employee that you received the assignment from (and them *only* since the others will be evaluating all of the exercises and should know as little as possible about which project belongs to you).

Assignment

For this project, you should implement as much of the following as you can in one week:

1. Start a new Clojure project using the leiningen build tool.
2. Pull in an HTTP library (ring is a common choice) and add a `/` HTTP endpoint that serves a headline and a nice "under construction" animated GIF. We love animated GIFs.
3. Add an `/about` HTTP endpoint that gives a brief description of the project.
4. Use a database of your choice to store TODO items (don't try to support multiple TODO *lists* yet).
    i. For the choice of database, SQLite and PostgreSQL are 2 common options. We use Datomic and welcome submissions using that, but it has a steep learning curve if you are new to it.
    ii. Each TODO item should keep track of its text and its doneness state (for now either `:todo` or `:done`).
5. Make the TODO items stored in the database show up on the `/` page in place of the "under construction" GIF you started with.
    i. Use the Hiccup library to render the HTML.
    ii. Add some dummy data to the database with at least 2 items in each doneness state.
    iii. Do *not* worry about making the items editable (even to mark them as `:done`) yet. We'll tackle that next.
6. Add a form to the `/` page for adding new TODO items to the list.
    i. The form should accept the text of the item, and when submitted, put it at the bottom of the list in `:todo` state.
7. Add a button next to each TODO item to "complete" it (mark it as `:done`).
    i. Change the button to "undo" marking it as complete when clicked and make it put that item back to `:todo` if clicked.
    ii. Make an item in `:done` state have its text struckthrough ~~like this~~.
8. Add a button next to each TODO item to delete it.
9. Modify the app to support multiple TODO lists (each with a name and their own set of TODO items).

10. From here you should get creative. Some ideas:
    i. Make it update in realtime across different browsers showing the same TODO lists.
    ii. Add user account support with data segration across accounts (i.e. when logged in I should only see *my* TODO lists).
    iii. Replace the server-rendered UI with a ClojureScript-based client-side UI (use your favorite React wrapper library; Om, Om.next, Reagent, etc.).

You should:

1. Use git for version control (we'll want the entire git repo submitted).
2. Make a commit that runs for each step above so we can see your progress as you built the project.
    i. Name these commits "Complete step 1", "Complete step 2", etc. so we can tell which they are.
    ii. Feel free to commit as often as you like besides that.